# About Partial Order Reduction in Planning and Computer Aided Verification

**Martin Wehrle**
University of Basel, Switzerland
University of Freiburg, Germany
martin.wehrle@unibas.ch

**Malte Helmert**
University of Basel, Switzerland
malte.helmert@unibas.ch

**Abstract**

Partial order reduction is a state space pruning approach that has been originally introduced in computer aided verification. Recently, various partial order reduction techniques have also been proposed for planning. Despite very similar underlying ideas, the relevant literature from computer aided verification has hardly been analyzed in the planning area so far, and it is unclear how these techniques are formally related.

We provide an analysis of existing partial order reduction techniques and their relationships. We show that recently proposed approaches in planning are instances of general partial order reduction approaches from computer aided verification. Our analysis reveals a hierarchy of dominance relationships and shows that there is still room for improvement for partial order reduction techniques in planning. Overall, we provide a first step towards a better understanding and a unifying theory of partial order reduction techniques from different areas.

## Introduction

Planning as heuristic search is one of the most successful approaches to domain-independent planning. However, as heuristics often estimate the distance to a goal state imperfectly, a general problem of heuristic search is the existence of plateaus in the state space. Therefore, additional pruning techniques are desirable for large state spaces. This is especially true for optimal planning, where recent results show that even almost perfect heuristics are not good enough to make the approach scalable (Helmert and Röger 2008).

Partial order reduction is an approach to reduce the size of the state space by avoiding a combinatorial blow-up induced by independent operators. Partial order reduction has first been proposed in the areas of Petri nets and computer aided verification. The basic idea is to reduce the size of the state space by selecting in each search state a subset of applicable transitions that is sufficient to preserve completeness. In this context, Valmari (1991) proposed the notion of *stubborn sets* $T_s$ with the property that transitions not occurring in such a set need not be applied immediately in the current state, but can also be applied later as they are independent of the transitions in $T_s$. Moreover, Godefroid (1991; 1996) proposed the notion of *sleep sets* with the property that transitions in such a set can be ignored because it is guaranteed that the corresponding states can still be generated through other paths. Sleep sets can have synergy effects with other partial order reduction techniques such as persistent sets (Godefroid 1996).

Recently, many techniques based on partial order reduction have also been proposed for planning. In particular, partial order reduction is the underlying idea of the techniques based on expansion cores (Chen and Yao 2009; Xu et al. 2011) and stratified planning (Chen, Xu, and Yao 2009; Xu et al. 2011). Moreover, commutativity pruning (Haslum and Geffner 2000) prunes paths that are redundant with other (commutative) paths. However, although these techniques are based on similar ideas, their formal relationships have not yet been adequately explored.

In this paper, we shed light on the relationships between partial order reduction techniques from planning and computer aided verification. As a benefit of our analysis, apart from a better theoretical understanding of existing techniques, we obtain insights about their pruning power and therefore, about which technique is likely to perform better in practice. As a side effect, our analysis also points us to problems with existing techniques and shows how these problems can be fixed. Furthermore, our analysis shows how existing partial order reduction techniques from planning can be improved. We give a short summary of the main results. The first part of the paper considers *state reduction* techniques. In this context, we show that (a corrected version of) the expansion core technique is an instance of *strong* stubborn sets. The second part of the paper considers *transition reduction* techniques. In this context, we show that sleep sets dominate commutativity pruning. Furthermore, we show that commutativity pruning dominates stratified planning, which is also a transition reduction technique rather than a state reduction technique as pointed out by the authors (Xu et al. 2011).

## Preliminaries

For a set $\mathcal{V}$ of finite-domain state variables, we define a *state* as a total assignment of values to the variables in $\mathcal{V}$. A *partial state* is defined as a partial assignment of values to variables in $\mathcal{V}$ (i. e., some variables can have an *undefined* value). We write $vars(s_p)$ for the set of variables for which partial state $s_p$ is defined. The notation $s \models s_p$ denotes that state $s$ agrees with $s_p$ on all variables in $vars(s_p)$.

**Definition 1** (**$SAS^+$ Planning Task).** An $SAS^+$ planning task (or planning task for short) is defined as a 4-tuple $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ with the following properties. $\mathcal{V}$ is a finite set of finite-domain state variables $v$ with domain $\mathcal{D}(v)$. $\mathcal{O}$ is a finite set of operators $o = \langle pre(o), \mathit{eff}(o)\rangle$ consisting of a *precondition pre* and an *effect eff*, where *pre* and *eff* are partial states. Moreover, $s_0$ is the *initial* state, and $s_\star$ is a partial state describing the goal states.

The value of a variable $v$ in a partial state $s$ is denoted with $s[v]$, i. e., $s[v] \in \mathcal{D}(v) \cup \{\mathit{undefined}\}$. For operators $o$, we define $\mathit{prevars}(o) = \mathit{vars}(pre(o))$ and $\mathit{effvars}(o) = \mathit{vars}(\mathit{eff}(o))$ as the sets of variables that $o$ reads and modifies, respectively. We also define the variable set of an operator as $\mathit{vars}(o) = \mathit{prevars}(o) \cup \mathit{effvars}(o)$. We assume that operators have non-empty effects (i. e., for all operators $o \in \mathcal{O}$, we have $\mathit{effvars}(o) \neq \emptyset$). To distinguish between preconditions and effects in example planning tasks, we write preconditions as constraints $v = d$, and effects as assignments $v := d$ for a variable $v$ and a value $d \in \mathcal{D}(v)$.

An operator $o$ is *applicable* in a state $s$ iff $s \models pre(o)$, i. e., iff the precondition of $o$ is consistent with $s$. In this case, the *successor state* $o(s)$ is defined as the state that is obtained from $s$ by changing the effect variables according to $\mathit{eff}(o)$ and retaining the other values from $s$. For a planning task $\pi$, a *solution* is defined as a sequence of operators that leads from $s_0$ to a state $s$ with $s \models s_\star$. More formally, $o_1, \dots, o_n$ is a solution if $o_n(o_{n-1}(\dots(o_1(s_0))\dots)) \models s_\star$, including the requirement that each operator is applicable in the intermediate state to which it is applied. We say that a variable $v$ is *goal-related* if $v \in \mathit{vars}(s_\star)$.

Moreover, we need the notion of *active states* and *active operators*. A state $s'$ is *active in state* $s$ if $s'$ is reachable from $s$ (i. e., there is a sequence of operators that leads from $s$ to $s'$) and a goal state is reachable from $s'$. An operator $o$ is *active in state* $s$ if there is an active state $s'$ in $s$ in which $o$ is applicable and such that $o(s')$ is active in $s$.

Furthermore, we need the notion of *domain transition graphs*. For a planning task $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ and variable $v \in \mathcal{V}$, the domain transition graph $DTG(v)$ is defined as a directed graph $(V, E)$ with $V = \mathcal{D}(v)$ such that there is an edge $e$ from $w$ to $w'$ iff there is an operator $o \in \mathcal{O}$ with $\mathit{eff}(o)[v] = w'$ and either $pre(o)[v] = w$ or $v \notin \mathit{prevars}(o)$. Note that $o$ induces an edge in $DTG(v)$ iff $v \in \mathit{effvars}(o)$.

Furthermore, we introduce different notions of operator dependencies that partial order reduction techniques rely on.

**Definition 2 (Dependencies Between Operators).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task. Let $o_1, o_2 \in \mathcal{O}$ be operators and let $s$ be a state of $\pi$. We say that

- $o_1$ *disables* $o_2$ in $s$ if $o_1$ and $o_2$ are both applicable in $s$, and $o_2$ is not applicable in $o_1(s)$,
- $o_1$ *enables* $o_2$ in $s$ if $o_1$ is applicable in $s$, $o_2$ is not applicable in $s$, and $o_2$ is applicable in $o_1(s)$,
- $o_1$ and $o_2$ *conflict* in $s$ if they are applicable in $s$ in either order, but the two orders result in different states,
- $o_1$ and $o_2$ *interfere* if there exists a state in which they conflict or one disables the other, and
- $o_1$ and $o_2$ are *commutative* if they do not interfere and neither enables the other in any state.

Informally, two operators $o$ and $o'$ do *not* interfere if for all states where both $o$ and $o'$ are applicable, their application in both possible orders is possible and leads to the same state. Specifically, $o$ and $o'$ do not interfere if there is no state where both $o$ and $o'$ are applicable. Commutativity is a stronger property than non-interference as it also requires operators not to enable one another.

We remark that in the $SAS^+$ formalism, interference and commutativity can be tested in polynomial time since the definition refers to syntactic states, not reachable or active states. We omit the technical details for brevity.

In the following, we discuss a number of techniques for reducing the search space of a planning task. We assume that the reader is familiar with planning based on heuristic forward search using algorithms such as A$^*$ and IDA$^*$. The objective of all discussed techniques is to reduce the search space of such algorithms while maintaining completeness of the search.

## State Reduction Techniques

*Stubborn sets* (Valmari 1991) and *expansion core* (Chen and Yao 2009; Xu et al. 2011) are partial order reduction techniques that have been proposed in the areas of Petri nets and planning, respectively. We briefly introduce these techniques and show that the expansion core technique is in fact an instance of *strong* stubborn sets.

### Stubborn Sets

*Stubborn sets* describe a partial order reduction technique that has been introduced in the area of Petri nets. The stubborn sets method only applies a subset of applicable operators in any search state, but chooses this subset in a way that preserves completeness of the search. The original approach has been proposed by Valmari (1991).

Valmari provides two variants of stubborn sets, *strong* stubborn sets and *weak* stubborn sets. Strong stubborn sets can be defined and implemented in an easier way than weak stubborn sets, but weak stubborn sets provide more pruning power. We introduce strong stubborn sets because they are most relevant for this paper. For the rest of this paper, when we refer to stubborn sets, the corresponding statement is true for both strong and weak stubborn sets.

Stubborn sets are defined over constraints that must hold for a set of operators. To compactly define these constraints, we first need the notion of *necessary enabling sets* for an operator $o$ and a state $s$. We adapt the definition of Godefroid (1996).

**Definition 3 (Necessary Enabling Set).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task. Let $s$ be a state, and let $o \in \mathcal{O}$ be an operator that is not applicable in $s$. A *necessary enabling set* for $o$ and $s$ is a set $N$ of operators such that all operator sequences $\sigma$ that include $o$ and lead from $s$ to some goal state contain some operator from $N$ before the first occurrence of $o$.

A necessary enabling set for $o$ and $s$ contains operators of which at least one must be applied before $o$. *Disjunctive action landmarks* (Helmert and Domshlak 2009) for the planning task $(\mathcal{V}, \mathcal{O}, s, pre(o))$ are examples of necessary enabling sets for $o$.[1] Strong stubborn sets can now be defined as follows. We adapt the definition of Godefroid (1996).

**Definition 4 (Strong Stubborn Set).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task, and let $s$ be a state. A set of operators $T_s \subseteq \mathcal{O}$ is called a *strong stubborn set* in $s$ if the following conditions hold.

1. For all operators $o \in T_s$ not applicable in $s$, $T_s$ contains a necessary enabling set for $o$ and $s$.
2. For all operators $o \in T_s$ applicable in $s$, $T_s$ contains all operators $o'$ that are active in $s$ and interfere with $o$.
3. $T_s$ contains a disjunctive action landmark for $(\mathcal{V}, \mathcal{O}, s, s_\star)$ (i. e., a set of operators including at least one operator from each path from $s$ to a goal state).[2]

Note that strong stubborn sets in a state $s$ may (and usually do) contain operators that are not applicable in $s$.

Apart from requirement 3, Def. 4 is slightly more permissive than the definition of Godefroid because for a given state $s$, we do not consider operators that are not active in $s$. For all operators $o_1, \dots, o_n$ that are active in $s$, strong stubborn sets $T_s$ have the following property. If $o_1, \dots, o_{n-1} \notin T_s$, $o_n \in T_s$, and $o_n(o_{n-1}(\dots(o_1(s))\dots))$ is defined, then $o_{n-1}(\dots o_1(o_n(s))\dots)$ is defined as well and leads to the same state. We further remark that, in contrast to the definition of Godefroid, Def. 4 does not explicitly require an (active) applicable operator to be included in $T_s$ if such an operator exists because this requirement already follows from our requirements 1. and 3.

When combined with optimal search algorithms, strong stubborn sets preserve optimality: for every optimal path to the goal that is pruned, a permutation of this path is not pruned. (We omit a formal proof due to space limitations.)

## Expansion Core (EC)

The *expansion core* technique (Chen and Yao 2009; Xu et al. 2011) is a partial order reduction technique that has been introduced in planning. It is based on reducing the number of applied operators in a state using *dependency closures* of variables. We give a brief introduction to this technique using a simplified notation.

**Definition 5 (Potential Precondition).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task, let $v, v' \in \mathcal{V}$ be variables, let $s$ be a state. Then $v$ is a *potential precondition* of $v'$ in $s$ if there is $o \in \mathcal{O}$ such that $pre(o)[v] = s[v]$, $v' \in \mathit{effvars}(o)$ and there is a path in $DTG(v')$ (to the goal vertex if $v'$ is goal-related) from $s[v']$ that includes an edge induced by $o$.

Informally, $v$ is a potential precondition of $v'$ in state $s$ if there is an operator $o$ that requires $v$ to have the value $pre(o)[v]$ in $s$ (i. e., $o$ reads $v$) and that modifies $v'$. To some extent, this is a state-dependent variant of dependency which extends the (state-independent) notion of dependency used in common definitions of causal graphs (e. g., Jonsson and Bäckström 1995; Williams and Nayak 1997). The additional requirement of the existence of a path in $DTG(v')$ strengthens the definition by ruling out some operators that cannot be applicable in any state reachable from $s$ or that lead to a dead end. More precisely, this condition is an overapproximation of active operators that rules out some, but not necessarily all operators that are not active in $s$. Furthermore, Chen and Yao give the definition of *potential dependency*.

**Definition 6 (Potential Dependent).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task, let $v, v' \in \mathcal{V}$ be variables, let $s$ be a state. Then $v$ is a *potential dependent* of $v'$ in $s$ if there is $o \in \mathcal{O}$ such that $v' \in \mathit{prevars}(o)$, $v \in \mathit{effvars}(o)$, $o$ induces an edge in $DTG(v)$ with source vertex $s[v]$, and there is a path in $DTG(v')$ from $s[v']$ (to the goal vertex if $v'$ is goal-related) that contains $pre(o)[v']$.

Informally, $v$ is a potential dependent of $v'$ in state $s$ if there is an operator $o$ that reads $v'$ and modifies $v$. The motivation to require $s[v]$ to be the source of an edge induced by $o$ in $DTG(v)$ is to rule out some operators that are not applicable in $s$. The motivation for the requirement of the existence of a path in $DTG(v')$ is analogous to Def. 5. Potential preconditions and dependents are related, but subtly different notions: $v'$ being a potential precondition of $v$ in $s$ does *not* imply that $v$ is a potential dependent of $v'$ in $s$, nor vice versa. Based on these notions, the *potential dependency graph* $PDG(s)$ in a state $s$ is defined as follows.

**Definition 7 (Potential Dependency Graph).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task, let $s$ be a state. The *potential dependency graph* $PDG(s)$ in $s$ is defined as the directed graph $(V, E)$ with $V = \mathcal{V}$ and edge set $E \subseteq V \times V$ with $(v, v') \in E$ iff $v \neq v'$ and $v$ is a potential precondition of $v'$, or $v$ is a potential dependent of $v'$, or there is $o \in \mathcal{O}$ with $\{v, v'\} \subseteq \mathit{effvars}(o)$.[3]

The EC algorithm is based on potential dependency graphs and works as follows. Given a non-goal state $s$, EC

[1]However, not all necessary enabling sets are of this form since they are defined with respect to *o and* the original goal.

[2]The original definition of stubborn sets applies to a setting without goals, where the objective is to find a dead-end state. It does not include this third requirement. However, goal reachability can be easily reduced to dead-end detection by introducing a few helper variables and operators. If we do this, the original requirements for the new operators lead to the new requirement 3.

[3]The last condition, forcing edges between variables modified by the same operator, extends the original definition of potential dependency graphs (Chen and Yao 2009; Xu et al. 2011). The original definition causes incompleteness of the expansion core technique due to missed interactions between effects. To see this, consider a task with $\mathcal{V} = \{a, b, c\}$, $\mathcal{O} = \{o_1, o_2\}$ with $o_1 = \langle c = 0; a := 1\rangle$ and $o_2 = \langle \top; b := 1, c := 1\rangle$, $s_0 \models (a = b = c = 0)$, and $s_\star \models (a = b = 1)$. The task has the solution $o_1 o_2$. With the original definition of PDGs, $\{b\}$ forms a dependency closure causing EC to only apply operator $o_2$ in $s_0$, causing a dead end.

The original paper (Chen and Yao 2009) uses a non-standard definition of domain transition graphs and defines operators slightly differently. The counterexample also applies there because "unknown" is not considered in the definition of *need(o)* used there.

first identifies a goal-related variable $v$ that does not yet have the required value (i. e., $s[v] \neq s_\star[v]$) and then computes a dependency closure $dc(s)$ (i. e., a subset $C$ of variables in $PDG(s)$ such that there is no edge from a variable in $C$ to a variable not in $C$) that contains $v$. The set of operators $expand(s)$ applied by the search algorithm in $s$ then contains exactly those operators $o$ which are applicable in $s$ and modify a variable in the dependency closure (i. e., for which $\mathit{effvars}(o) \cap dc(s) \neq \emptyset$).

## Expansion Core Instantiates Strong Stubborn Sets

In this section, we show that the expansion core technique is in fact an instance of strong stubborn sets. In contrast to the work of Xu et al. (2011), we address the *corrected* version of the expansion core technique based on Def. 7 (the originally proposed EC technique cannot produce stubborn sets; see Footnote 3).

For a state $s$, a dependency closure $dc(s)$ and an operator $o$, we write $o \in dc(s)$ iff $\mathit{effvars}(o) \cap dc(s) \neq \emptyset$. To see that EC is an instance of strong stubborn sets, we need the following definition. For a state $s$, we define $DTGactive(s)$ as an overapproximation of the set of operators that are active in $s$. The overapproximation is based on domain transition graphs as suggested by the EC technique.

**Definition 8 (DTGactive).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task and let $s$ be a state. $DTGactive(s)$ is the set of operators $o \in \mathcal{O}$ that satisfy the following conditions:

1. For every variable $v \in prevars(o)$, there exists a path in $DTG(v)$ from $s[v]$ to $pre(o)[v]$.
2. For every goal-related variable $v$, if $v \in \mathit{effvars}(o)$, then there exists a path in $DTG(v)$ from $\mathit{eff}(o)[v]$ to the goal value $s_\star[v]$.

This definition captures the path existence criterion in Def. 5 and Def. 6 to rule out some operators that are not active. More precisely, we obtain an overapproximation of the operators that are needed to reach a goal state from $s$: operators that do not belong to $DTGactive(s)$ cannot be applied in any state reachable from $s$, or they lead to a state from which no goal is reachable. In other words, operators that do not belong to $DTGactive(s)$ are not active in $s$, but not vice versa. The definition of $DTGactive(s)$ is needed to show the following lemma. It states that, under certain conditions, operators that work on common variables "belong to" the same dependency closure. As a side remark, this would *not* hold under the original definition of potential dependency graphs.

**Lemma 1.** *Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task. Let $s$ be a state and $dc(s)$ be a dependency closure. Let $o, o' \in \mathcal{O}$ be operators for which at least one of the following conditions holds:*

1. *$o$ and $o'$ modify a common variable.*
2. *$o$ modifies a variable that $o'$ reads, and $s[v] = pre(o')[v]$ for such a variable $v$.*
3. *$o$ reads a variable that $o'$ modifies, and $o$ is applicable in $s$.*

*If $o, o' \in DTGactive(s)$ and $o \in dc(s)$, then $o' \in dc(s)$.*

*Proof.* We distinguish the following three cases.

1. $o$ and $o'$ modify a common variable. Let $v'$ be such a variable. Since $o \in dc(s)$, there exists a variable $v \in \mathit{effvars}(o) \cap dc(s)$. Because all variables modified by the same operator are connected in $PDG(s)$ and $dc(s)$ is closed under outgoing edges of $PDG(s)$, this implies that $v' \in dc(s)$ and hence $v' \in \mathit{effvars}(o') \cap dc(s)$, which proves $o' \in dc(s)$.
2. $o$ modifies a variable $v$ that $o'$ reads, i. e., $v \in \mathit{effvars}(o) \cap prevars(o')$, and $s[v] = pre(o')[v]$ for this variable. Consider a variable $v' \in \mathit{effvars}(o')$ (such a variable must exist because operators have non-empty effects). We can assume $v \neq v'$; otherwise Case 1 applies. We show that $v$ is a potential precondition of $v'$. There is an edge $e$ in $DTG(v')$ that is induced by $o'$ such that there is a path in $DTG(v')$ (that reaches the goal vertex if $v'$ is goal-related) from $s[v']$ that includes $e$. To see this, consider the following cases.
   - If $v' \in prevars(o')$, then there is a path from $s[v']$ to $pre(o')[v']$ in $DTG(v')$ because $o' \in DTGactive(s)$. Therefore, there is a path from $s[v']$ that includes $e$. If $v'$ is goal-related, then there is a path from $\mathit{eff}(o')[v']$ to $s_\star[v']$ because $o' \in DTGactive(s)$. Therefore, in this case, there is a path from $s[v']$ to $s_\star[v']$ that includes $e$.
   - If $v' \notin prevars(o')$, then by definition of domain transition graphs, $o'$ induces an edge from *every* node in $DTG(v')$ to $\mathit{eff}(o')[v']$. Therefore, trivially there is a path from $s[v']$ that includes $e$, namely the path from $s[v']$ to $\mathit{eff}(o)[v']$. If $v'$ is goal-related, then there is a path from $\mathit{eff}(o)[v']$ to $s_\star[v']$ because $o' \in DTGactive(s)$. Therefore, in this case, there is a path from $s[v']$ to $s_\star[v']$ that includes $e$.

   Overall, we observe that $s[v] = pre(o')[v]$, $v' \in \mathit{effvars}(o')$, and there is a (goal related) path in $DTG(v')$ from $s[v']$ that includes an edge induced by $o'$. Therefore, $v$ is a potential precondition of $v'$ and hence, there is an edge in $PDG(s)$ from $v$ to $v'$. From $v \in dc(s)$ and the closure of $dc(s)$ we get $v' \in dc(s)$ and hence $o' \in dc(s)$.
3. $o$ reads a variable that $o'$ modifies, i. e., there is a variable $v \in prevars(o) \cap \mathit{effvars}(o')$. Let $v'$ be a variable with $v' \in \mathit{effvars}(o)$ (such a variable must exist because operators have non-empty effects). As in Case 2 we can assume $v \neq v'$. We show that $v'$ is a potential dependent of $v$. First, according to our assumption, $v \in prevars(o)$. Furthermore, we observe that there is an edge $e = (s[v'], \mathit{eff}(o)[v'])$ in $DTG(v')$ because $v' \in \mathit{effvars}(o)$. To see this, consider the following cases.
   - $v' \in prevars(o)$. Then $pre(o)[v'] = s[v']$ because $o$ is applicable in $s$, and therefore, there is the edge $(pre(o)[v'], \mathit{eff}(o)[v'])$ induced by $o$ in $DTG(v')$.
   - $v' \notin prevars(o)$. Then there is an edge in $DTG(v')$ from *every* vertex in $DTG(v')$ to $\mathit{eff}(o)[v']$, specifically from the vertex $s[v']$.

   Moreover, there trivially exists a path in $DTG(v)$ from $s[v]$ to $pre(o)[v]$, namely the empty path because $s[v] = pre(o)[v]$. If $v$ is goal-related, then there is also a path from $\mathit{eff}(o)[v]$ to the goal vertex in $DTG(v)$ because $o \in$

$DTGactive(s)$. Overall, this shows that $v'$ is a potential dependent of $v$, and therefore, there is an edge from $v'$ to $v$ in $PDG(s)$. From $o \in dc(s)$ and $v' \in \mathit{effvars}(o)$ we get $v' \in dc(s)$; due to the edge from $v'$ to $v$ we get $v \in dc(s)$; with $v \in \mathit{effvars}(o')$ we get $o' \in dc(s)$. □

The main theorem of this section shows that, for a state $s$, the set of operators to be applied in $s$ identified with the expansion core technique offers the same pruning power as a strong stubborn set. To see this, we consider the set $EC(s) = expand(s) \cap DTGactive(s)$ consisting of the set of operators identified with the expansion core technique that are also in $DTGactive(s)$. Again, note that excluding operators that are not in $DTGactive(s)$ (i. e., in this case, some operators that lead to a state from which no goal state is reachable) does not change the goal-reachable state space from $s$. Considering $EC(s)$ instead of $expand(s)$ is a straight-forward optimization of the expansion core technique, which has not been used in the original approach although the basic ideas have already been given there.

In the following, we construct a strong stubborn set $T_s$ with the same pruning power as $EC(s)$. Before describing the construction, we prove the following lemma.

**Lemma 2.** *Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task, $s$ be a state. Let $dc(s)$ be the dependency closure that is selected by the EC reduction technique in $s$. Let $o$ be an operator such that $o \in DTGactive(s)$, $o \in dc(s)$, and $o$ is not applicable in $s$. Then*

$$en(o,s) = \{o' \mid o' \in dc(s), o' \in DTGactive(s), \exists v : \\ \mathit{eff}(o')[v] = pre(o)[v] \neq \mathit{undefined}\}$$

*is a necessary enabling set for $o$ and $s$.*

*Proof.* We show that for every sequence $\sigma = o_1, \dots, o_n$ of operators that leads from $s$ to a state $s'$ where $o$ is applicable and from which a goal state is reachable, there is an operator $o_i \in \{o_1, \dots, o_n\}$ such that $o_i \in en(o,s)$. Consider such a sequence of operators $\sigma = o_1, \dots, o_n$. First, we observe that all operators in $\sigma$ are also in $DTGactive(s)$ because they are applied in $\sigma$ and lead to a state from which a goal state is reachable. Furthermore, we observe that for every variable $v$ with $v \in prevars(o)$ and $pre(o)[v] \neq s[v]$, there is an operator $o_i \in \{o_1, \dots, o_n\}$ such that $\mathit{eff}(o_i)[v] = pre(o)[v]$ (otherwise, $o$ cannot be applicable after $\sigma$ is applied). Consider such a variable $v$ and such an operator $o_i$. We distinguish two cases.

1. $v \in \mathit{effvars}(o)$. Then $o$ and $o_i$ both modify $v$. Hence, $o_i \in dc(s)$ because $o \in dc(s)$. Therefore, $o_i \in en(o,s)$.
2. $v \notin \mathit{effvars}(o)$. Then there is another variable $v'$ such that $v' \in \mathit{effvars}(o)$ because operators have non-empty effects. We distinguish the following two cases.
   (a) $v' \in prevars(o)$ and $pre(o)[v'] \neq s[v']$. Then there is $o_j$ in $\sigma$ such that $\mathit{eff}(o_j)[v'] = pre(o)[v']$. We observe that $o_j$ and $o$ both modify $v'$. Hence, $o_j \in dc(s)$ because $o \in dc(s)$. Therefore, $o_j \in en(o,s)$.
   (b) Otherwise, $v'$ is a potential dependent of $v$ and we have an edge in $PDG(s)$ from $v'$ to $v$. As $v' \in \mathit{effvars}(o)$ and $v \in \mathit{effvars}(o_i)$, we have that $o_i \in dc(s)$. Therefore, $o_i \in en(o,s)$.

The above reasoning shows that it suffices to consider enabling operators from the same dependency closure to obtain necessary enabling sets. Therefore, $en(o,s)$ is a necessary enabling set for $o$ and $s$. □

We are now ready to construct a strong stubborn set $T_s$ with the same pruning power as $EC(s)$.

**Theorem 1.** *Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task, let $s$ be a non-goal state, and let $EC(s) = expand(s) \cap DTGactive(s)$. Let*

$$T_s^{\text{dc}} := \{o \in \mathcal{O} \mid o \in dc(s), o \in DTGactive(s)\}$$
$$T_s^{\text{int}} := \{o' \in \mathcal{O} \mid o' \textit{ interferes with } o \in EC(s), \\ o' \in DTGactive(s)\} \setminus T_s^{\text{dc}}.$$

*Then $T_s := T_s^{\text{dc}} \cup T_s^{\text{int}}$ is a strong stubborn set in $s$, and the set of operators in $T_s$ that are applicable in $s$ is equal to $EC(s)$.*

*Proof.* We first observe that all $o' \in T_s^{\text{int}}$ are inapplicable in $s$. To see this, assume that $o' \in T_s^{\text{int}}$ were applicable. Let $o \in EC(s)$ be an operator it interferes with. By definition of $EC(s)$, $o$ is applicable. Moreover, $o, o' \in DTGactive(s)$ and $o \in dc(s)$, and because $o$ and $o'$ interfere, one of the three cases of Lemma 1 applies. Thus we get $o' \in dc(s)$, which implies $o' \in T_s^{\text{dc}}$, contradicting $o' \in T_s^{\text{int}}$.

Since all operators in $T_s^{\text{int}}$ are inapplicable, the applicable operators of $T_s$ are the applicable operators of $T_s^{\text{dc}}$, which are exactly the operators $EC(s)$, proving the second part of the theorem.

To prove the first part of the theorem, we have to verify the three conditions for strong stubborn sets (Def. 4).

The second condition is easy to show: the applicable operators in $T_s$ are $EC(s)$, and $T_s$ clearly contains all active operators interfering with operators in $EC(s)$.

The third condition is also easy: by construction $dc(s)$ contains a variable which is not yet at goal value, and $T_s^{\text{dc}}$ contains all active operators that can change this variable. These operators form a disjunctive action landmark.

To show the first condition, we must show that $T_s$ contains a necessary enabling set for all inapplicable operators $o \in T_s$ and $s$. If $o \in T_s^{\text{dc}}$, Lemma 2 implies that $T_s^{\text{dc}} \subseteq T_s$ is such a necessary enabling set for $o$ and $s$.

We conclude the proof by showing that for $o' \in T_s^{\text{int}}$, $T_s^{\text{dc}}$ is also a necessary enabling set in $s$. Let $o'$ be such an operator, and let $o \in EC(s)$ be an operator it interferes with. Because $o \in dc(s)$, $\mathit{effvars}(o) \subseteq dc(s)$. (This is true for at least one effect variable by definition of $o \in dc(s)$ and for the others because effect variables of the same operator are connected in $PDG(s)$.) Because $o$ is applicable in $s$ and contained in $DTGactive(s)$, Def. 6 then implies that also $prevars(o) \subseteq dc(s)$, and hence $vars(o) \subseteq dc(s)$. There are three possible reasons why $o'$ could interfere with $o$: $o'$ and $o$ may modify a common variable $v$; $o'$ may modify a variable $v$ read by $o$; or $o'$ may read a variable $v$ modified by $o$. Because $vars(o) \subseteq dc(s)$, in all three cases we have $v \in dc(s)$. Hence, in the first two cases we immediately get that $o'$ modifies a variable in $dc(s)$, which implies $o' \in T_s^{\mathrm{dc}}$, a contradiction. The remaining case is that $o'$ reads a variable $v \in dc(s)$ which $o$ modifies. If $s[v] = pre(o')[v]$, then by Lemma 1 (part 2) we again get $o' \in dc(s)$ leading to contradiction. Therefore, we must have $s[v] \neq pre(o')[v]$, which means that $v$ must change for $o'$ to become applicable. Because $v \in dc(s)$, all active operators changing $v$ are in $T_s^{\mathrm{dc}}$, and thus $T_s^{\mathrm{dc}}$ is a necessary enabling set for $o'$, which concludes the proof. □

The theorem shows that in a state $s$, the operators in $s$ identified with a corrected version (and a straight-forward optimization) of EC are the same as the applicable operators identified with a strong stubborn set. Hence, the pruning power of strong stubborn sets is at least as high as the pruning power of EC. We remark that, as shown by Godefroid, reducing strong stubborn sets $S$ to the applicable operators in $S$ yields *persistent* sets (Godefroid 1996). Hence, Theorem 1 also shows that the expansion core method identifies *persistent sets*.

## Transition Reduction Techniques

In this section we consider partial order reduction techniques that reduce the number of *state transitions* considered during search, but do not reduce the number of reachable *states*. Specifically, we consider pruning methods based on *sleep sets* (Godefroid 1991; 1996), commutativity pruning (Haslum and Geffner 2000), and stratified planning (Chen, Xu, and Yao 2009; Xu et al. 2011). Sleep sets have been proposed in the area of computer aided verification, whereas commutativity pruning and stratified planning have been proposed in the area of planning.[4]

The pruning decisions of these algorithms are path-dependent: whether a given applicable operator $o$ is explored in state $s$ depends not just on $s$ but also on the search path via which $s$ was reached. Hence these techniques are not immediately applicable to graph search algorithms performing duplicate elimination such as A$^*$. However, they can be very useful in the context of tree search algorithms such as IDA$^*$ (e. g., Haslum and Geffner 2000). Due to the dependency on paths, in this section we must carefully distinguish between *states* $s$ of the world and *paths* $\sigma$ (operator sequences originating in the initial state). Tree search algorithms search in the space of paths, starting from the empty path $\epsilon$. Where this makes sense, we apply state terminology to paths: for example, an operator is applicable in path $\sigma$ if it is applicable in the state reached after following path $\sigma$.

[4]All pruning techniques in this section are *goal-independent*, i. e., make their pruning choices without considering the goal $s_\star$ of the planning task. Goal-independent pruning algorithms cannot prune reachable states without losing completeness: if they pruned a reachable state $s$, then they would be incomplete for a modified task where $s_\star = s$. Stratified planning, one of the techniques we consider, was originally motivated as a technique for reducing the number of reachable states (Chen, Xu, and Yao 2009), but this is corrected in a later publication (Xu et al. 2011).

We show that the sleep sets technique strictly dominates commutativity pruning. Furthermore, we show that commutativity pruning strictly dominates stratified planning.

### Sleep Sets Dominate Commutativity Pruning

*Sleep sets* (Godefroid 1991; 1996) is a partial order reduction technique from computer aided verification. Transformed to the planning setting, a sleep set for a path $\sigma$ is a set of operators that are applicable after path $\sigma$ but skipped during search. Godefroid provides a simple algorithm to compute sleep sets on the fly. We follow the version of the sleep set algorithm presented in his monograph (Godefroid 1996), simplified for a setting without duplicate elimination:

1. Search begins with an empty sleep set: $sleep(\epsilon) := \emptyset$.
2. When expanding path $\sigma$, only generate successors for the applicable operators that are not in $sleep(\sigma)$. Let $o_1, \dots, o_n$ be these operators, ordered in the same way as they are considered by the search algorithm. For each successor path $\sigma o_i$, set $sleep(\sigma o_i) := (sleep(\sigma) \cup \{o_1, \dots, o_{i-1}\}) \setminus \{o \mid o \text{ and } o_i \text{ are not commutative}\}$.

*Commutativity pruning* (Haslum and Geffner 2000) is a simple partial order reduction technique that has been introduced in AI planning in the context of IDA$^*$ search. The idea is to impose an (arbitrary) total order $<_c$ on the operators, and to subsequently apply commutative operators only if the order of application is consistent with $<_c$. More precisely, when expanding a path $\sigma$ whose last operator is $o$, a successor path $\sigma o'$ is only generated if $o <_c o'$ or $o$ and $o'$ are not commutative. The following considerations show that under suitable ordering choices, sleep sets prune all paths pruned by commutativity pruning.

**Proposition 1.** *Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task, and let $<$ be the total order on $\mathcal{O}$ used for commutativity pruning. If the sleep set method considers the applicable operators for each path in the order defined by $<$, then all paths pruned by commutativity pruning are also pruned by the sleep set method.*

*Proof.* First, note that with the given operator ordering for the sleep set method, the definition of sleep sets for successor paths can be rewritten as $sleep(\sigma o_i) = (sleep(\sigma) \cup \{\hat{o} \mid \hat{o} < o_i \text{ and } \hat{o} \text{ applicable in } \sigma\}) \setminus \{\hat{o} \mid \hat{o} \text{ and } o_i \text{ are not commutative}\}$.

Consider a path $\tau$ pruned by commutativity pruning (i. e., the path itself is not generated, but its "parent" path, consisting of $\tau$ without the last operator, is generated). Then $\tau$ is of the form $\tau = \sigma o o'$ where $\sigma$ is some path and $o$ and $o'$ are commutative operators satisfying $o' < o$.[5]

Consider the behaviour of the sleep sets technique in $\sigma$. Because the path $\sigma o o'$ was considered, it follows that $o$ is applicable in $\sigma$ and $o'$ is applicable in $\sigma o$. Because the two operators are commutative, this implies that $o'$ is also applicable in $\sigma$. From the rewritten definition of sleep sets, we get $sleep(\sigma o) = (sleep(\sigma) \cup \{\hat{o} \mid \hat{o} < o \text{ and } \hat{o} \text{ applicable in } \sigma\}) \setminus \{\hat{o} \mid$

[5]We assume that all search algorithms prune operators that do not change the state, and hence we can ignore the case $o = o'$.

$\hat{o}$ and $o$ are not commutative $\}$. This set contains $o'$ since $o' < o$ and the two operators are commutative. We conclude that $o' \in \mathit{sleep}(\sigma o)$, which means that the sleep sets technique also prunes the path $\tau = \sigma o o'$. $\square$

We now show that the dominance of sleep sets over commutativity pruning based on the same total order is strict: there exist cases where some paths are pruned by sleep sets but not by commutativity pruning.

**Example 1.** *Consider a planning task with variables $\mathcal{V} = \{a, b, c\}$. All variables have domain $\{0, 1\}$ and are initially set to $0$. There exist three operators: $o_1 = \langle a = 1; a := 0 \rangle$, $o_2 = \langle c = 0; c := 1 \rangle$ and $o_3 = \langle b = 0; a := 1, b := 1 \rangle$. The operators are ordered $o_1 < o_2 < o_3$.*

*The path $o_3 o_1 o_2$ is pruned by the sleep set method because all paths $\tau$ starting with $o_3$ satisfy $o_2 \in \mathit{sleep}(\tau)$. However, it is not pruned by commutativity pruning: the prefix $o_3 o_1$ is not pruned because the two operators are not commutative, and appending $o_2$ after $o_1$ is allowed because $o_1 < o_2$.*

Overall, Prop. 1 and Ex. 1 show that sleep sets offer more pruning power than commutativity pruning.

**Theorem 2.** *Sleep sets dominate commutativity pruning.*

Under the same ordering of operators, all paths pruned by commutativity pruning are also pruned by sleep sets, while the converse is not true.

## Commutativity Pruning Dominates Stratified Planning

In this section, we show that commutativity pruning dominates stratified planning in the sense that more paths are pruned with commutativity pruning than with stratified planning. The considerations in this section refer to the *updated* version of stratified planning (Xu et al. 2011), not to the original version (Chen, Xu, and Yao 2009).[6] We first give a short description of stratified planning.

**Definition 9 (Causal Graph).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task. The *causal graph* of $\pi$ is defined as the directed graph $CG = (V, E)$ with $V = \mathcal{V}$ which has the edge $(v, v') \in E$ iff $v \neq v'$ and there exists an operator $o \in \mathcal{O}$ such that $v \in \mathit{vars}(o)$ and $v' \in \mathit{effvars}(o)$.

Based on causal graphs, every variable $v \in \mathcal{V}$ is assigned a *level*. Let $C_1, \dots, C_n$ be the strongly connected components of $CG$ in a topological ordering such that edges from variables in $C_i$ to variables in $C_j$ only exist if $i \leq j$. Then $\mathit{level}(v) = i$ iff $v \in C_i$.[7] Furthermore, levels are also assigned to operators $o \in \mathcal{O}$ according to the levels of the variables in their effects: $\mathit{level}(o) = i$ iff $v \in \mathit{effvars}(o)$ and $\mathit{level}(v) = i$. This is well-defined because Def. 9 implies that variables that occur as effects of the same operator have the same level. Stratified planning also uses the notion of *follow-up operators*.

[6]The original version (Chen, Xu, and Yao 2009) contains an error due to the non-standard definition of causal graphs; this error has been corrected in the updated version (Xu et al. 2011). We also remark that edge directions in the causal graphs we use here are inverted compared to the papers on stratified planning. Our definition follows the usage in the wider planning literature.

[7]Xu et al. allow combining multiple strongly connected components into a single level, but this only serves to reduce the pruning power of the approach.

**Definition 10 (Follow-up Operator).** Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task, and let $o$ and $o'$ be operators. Then $o'$ is a *follow-up operator* of $o$ if $\mathit{vars}(o') \cap \mathit{effvars}(o) \neq \emptyset$ (i. e., $o'$ reads or modifies some variable that $o$ modifies).

The stratified planning algorithm works as follows. When expanding a path $\sigma o$ in which operator $o'$ is applicable, the path $\sigma o o'$ is pruned if $\mathit{level}(o') > \mathit{level}(o)$ and $o'$ is not a follow-up operator of $o$.

In the following, we show that stratified planning is dominated by commutativity pruning. To see this, we first show that an operator that is pruned by stratified planning and the previous operator are commutative. This is stated in the following lemma.

**Lemma 3.** *Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task. Let $\sigma o o'$ be a path pruned by stratified planning. Then $o$ and $o'$ are commutative.*

*Proof.* Let $\sigma o o'$ be a path pruned by stratified planning. This implies $\mathit{level}(o') > \mathit{level}(o)$ and $o'$ is not a follow-up operator of $o$. The latter condition means that $o'$ does not read or modify a variable modified by $o$. Therefore, $o$ and $o'$ can only be non-commutative if $o$ reads some variable modified by $o'$. We assume that this is the case and derive a contradiction.

Let $v' \in \mathit{prevars}(o) \cap \mathit{effvars}(o')$. (By the assumption, such a variable exists.) Also, let $v \in \mathit{effvars}(o)$. (Such a variable exists because we required that all operators have non-empty effects.) We must have $v \neq v'$; otherwise $o'$ would be a follow-up operator of $o$. From $v' \in \mathit{prevars}(o)$ and $v \in \mathit{effvars}(o)$ we get that $(v', v)$ is an edge in the causal graph, which by the definition of levels implies $\mathit{level}(v') \leq \mathit{level}(v)$. From $v' \in \mathit{effvars}(o')$ we get $\mathit{level}(v') = \mathit{level}(o')$; similarly $\mathit{level}(v) = \mathit{level}(o)$. Putting things together, we have $\mathit{level}(o') = \mathit{level}(v') \leq \mathit{level}(v) = \mathit{level}(o)$. This contradicts $\mathit{level}(o') > \mathit{level}(o)$, concluding the proof. $\square$

We now show that for suitable operator orders $<_c$, all paths pruned by stratified planning are also pruned by commutativity pruning.

**Proposition 2.** *Let $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ be a planning task. Given a definition of stratified planning operator levels for $\mathcal{O}$, define a total operator order $<_c$ in such a way that $\mathit{level}(o) > \mathit{level}(o')$ implies $o <_c o'$. (Operators within the same level may be ordered arbitrarily by $<_c$.)*

*Then all paths pruned by stratified planning are pruned by commutativity pruning using the order $<_c$.*

*Proof.* Let $\tau$ be a path pruned by stratified planning. Then $\tau$ is of the form $\tau = \sigma o o'$ where $\mathit{level}(o') > \mathit{level}(o)$ and hence $o' <_c o$. By the previous lemma, $o$ and $o'$ are commutative. Together, these conditions mean that $\tau$ is also pruned by commutativity pruning based on $<_c$. $\square$

Finally, we show that commutativity pruning can prune operators that are not pruned by stratified planning.

**Example 2.** *Consider a planning task $\pi = (\mathcal{V}, \mathcal{O}, s_0, s_\star)$ with $\mathcal{V} = \{a, b, c, d\}$, operators $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$ with $o_1 = \langle a = 0; b := 1\rangle$, $o_2 = \langle c = 0; d := 1\rangle$, $o_3 = \langle d = 1; a := 1\rangle$ and $o_4 = \langle b = 1; c := 1\rangle$ and initial state $s_0 \models (a = b = c = d = 0)$. First, we observe that the causal graph is strongly connected. Therefore, all operators have the same level, and stratified planning cannot prune any paths. Now consider commutativity pruning with the order $o_1 <_c o_2 <_c o_3 <_c o_4$. (This is compatible with the requirements in Prop. 2 because operators with the same level can be ordered arbitrarily). We observe that, for example, commutativity pruning prunes the path $o_2 o_1$.*

Overall, Prop. 2 and Ex. 2 show that commutativity pruning can prune more operators than stratified planning.

**Theorem 3.** *Commutativity pruning dominates stratified planning.*

In summary, stratified planning can be seen as an instance of commutativity pruning except for operators with equal levels. For operators with equal levels, stratified planning does not offer any pruning power.

We conclude the section with a brief discussion on optimality. When combined with optimal search algorithms, sleep sets preserve optimality: for every optimal path to the goal that is pruned, a permutation of this path is not pruned. (We omit a formal proof due to space limitations.) It follows that commutativity pruning and stratified planning are optimality preserving as well.

## Related Work

Partial order reduction techniques in computer aided verification have been proposed based on stubborn sets (Valmari 1991), persistent sets (Godefroid 1996), and ample sets (Peled 1993). Stubborn sets in their original form preserve deadlocks. In contrast to strong stubborn sets, *weak* stubborn sets (Valmari 1991) $T_s^{\mathrm{w}}$ allow some operators that interfere with applicable operators from $T_s^{\mathrm{w}}$ to be excluded from $T_s^{\mathrm{w}}$. This is a less restrictive condition than the condition for strong stubborn sets. The concept of *ample sets* extends the original definition of stubborn and persistent sets such that they can be used for model checking $LTL_{-X}$ (i. e., linear temporal logical without the "next" operator $X$). Such extensions have also been proposed for strong stubborn sets (Valmari 1991; 1992).

There are various algorithms and heuristics how to actually compute stubborn sets (Valmari 1992; Godefroid 1996; Geldenhuys, Hansen, and Valmari 2009), persistent sets (Godefroid 1996), and ample sets (Clarke, Grumberg, and Peled 2000). In particular, ample sets have also been applied for directed model checking (Edelkamp, Leue, and Lluch-Lafuente 2004). Moreover, recent work also includes investigations about the quality of the reduced sets and the resulting state space reductions (Geldenhuys, Hansen, and Valmari 2009; Valmari and Hansen 2010).

These approaches are also relevant for planning, but have not been studied in depth in this context. To the best of our knowledge, the only work in planning that relates partial order approaches from computer aided verification and planning is the paper by Xu et al. (2011). Xu et al. define a set as stubborn if certain non-interference properties hold. These properties follow from the definition of strong stubborn sets as shown by Valmari (1991); as far as we know, the converse relationship has not been studied.

## Conclusion

Partial order reduction is an established approach to tackle the state explosion problem in computer aided verification and planning. We have shown that the techniques from computer aided verification are at least as powerful as the techniques proposed in planning. Specifically, we have observed that the pruning power of the expansion core technique is at most as high as the pruning power of strong stubborn sets, for which other algorithms have been suggested (as discussed in the related work section). Moreover, this result suggests that more pruning power is achievable when considering weak stubborn sets. Our analysis also pointed us to a problem with the expansion core technique and how this problem can be fixed. Furthermore, we have observed that sleep sets offer strictly more pruning power than commutativity pruning, which in turn offers strictly more pruning power than stratified planning.

Overall, we learned that from a planning point of view, we should not try to reinvent the wheel (which can be tedious and error-prone), but rather build on and further develop existing, well-established techniques. As a starting point, one could adapt and evaluate the algorithms suggested in the computer aided verification literature to compute stubborn and ample sets in the planning setting. Many of these algorithms could be adapted in a rather straight-forward way. Moreover, there is the question whether existing algorithms can be specialized for (classical) planning to obtain more pruning power. For example, in planning we do not have to preserve stuttering equivalence, which is needed for $LTL_{-X}$ model checking (Clarke, Grumberg, and Peled 2000). Finally, it would be interesting to investigate the relationships to other recent partial order reduction techniques from planning such as bounded intention planning (Wolfe and Russell 2011) and move pruning (Burch and Holte 2011).

## Acknowledgments

We thank the anonymous reviewers for their comments, which helped improve this paper.

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems". See `http://www.avacs.org/` for more information.

## References

Burch, N., and Holte, R. C. 2011. Automatic move pruning in general single-player games. In Borrajo, D.; Likhachev, M.; and López, C. L., eds., *Proceedings of the Fourth Annual Symposium on Combinatorial Search (SOCS 2011)*, 31–38. AAAI Press.

Chen, Y., and Yao, G. 2009. Completeness and optimality preserving reduction for planning. In Boutilier, C., ed., *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11–17, 2009*, 1659–1664.

Chen, Y.; Xu, Y.; and Yao, G. 2009. Stratified planning. In Boutilier, C., ed., *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11–17, 2009*, 1665–1670.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 2000. *Model Checking*. The MIT Press.

Edelkamp, S.; Leue, S.; and Lluch-Lafuente, A. 2004. Partial-order reduction and trail improvement in directed model checking. *International Journal on Software Tools for Technology Transfer* 6(4):277–301.

Geldenhuys, J.; Hansen, H.; and Valmari, A. 2009. Exploring the scope for partial order reduction. In Liu, Z., and Ravn, A. P., eds., *Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis (ATVA 2009)*, volume 5799 of *LNCS*, 39–53. Springer-Verlag.

Godefroid, P. 1991. Using partial orders to improve automatic verification methods. In Clarke, E. M., and Kurshan, R. P., eds., *Proceedings of the 2nd International Workshop on Computer Aided Verification (CAV 1990)*, volume 531 of *LNCS*, 176–185. Springer-Verlag.

Godefroid, P. 1996. *Partial-Order Methods for the Verification of Concurrent Systems — An Approach to the State-Explosion Problem*, volume 1032 of *LNCS*. Springer-Verlag.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In Chien, S.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, 140–149. AAAI Press.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M., and Röger, G. 2008. How good is almost perfect? In Fox, D., and Gomes, C. P., eds., *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, 944–949. AAAI Press.

Jonsson, P., and Bäckström, C. 1995. Incremental planning. In Ghallab, M., and Milani, A., eds., *New Directions in AI Planning: EWSP '95 — 3rd European Workshop on Planning*, volume 31 of *Frontiers in Artificial Intelligence and Applications*, 79–90. Amsterdam: IOS Press.

Peled, D. 1993. All from one, one for all: on model checking using representatives. In Courcoubetis, C., ed., *Proceedings of the 5th International Conference on Computer Aided Verification (CAV 1993)*, volume 697 of *LNCS*, 409–423. Springer-Verlag.

Valmari, A., and Hansen, H. 2010. Can stubborn sets be optimal? In Lilius, J., and Penczek, W., eds., *Proceedings of the 31st International Conference on Applications and Theory of Petri Nets (Petri Nets 2010)*, volume 6128 of *LNCS*, 43–62. Springer-Verlag.

Valmari, A. 1991. Stubborn sets for reduced state space generation. In Rozenberg, G., ed., *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets (APN 1989)*, volume 483 of *LNCS*, 491–515. Springer-Verlag.

Valmari, A. 1992. A stubborn attack on state explosion. *Formal Methods in System Design* 1(4):297–322.

Williams, B. C., and Nayak, P. P. 1997. A reactive planner for a model-based executive. In Pollack, M. E., ed., *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI 1997)*, 1178–1185. Morgan Kaufmann.

Wolfe, J., and Russell, S. J. 2011. Bounded intention planning. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, 2039–2045. AAAI Press/IJCAI.

Xu, Y.; Chen, Y.; Lu, Q.; and Huang, R. 2011. Theory and algorithms for partial order based reduction in planning. *CoRR* abs/1106.5427.